

# Homework Assignment-- NextDate-- test case design

by:

Abdullah Hanoosh (100749026)

Github: <https://github.com/abduh3131/SQ-Delv-Homwork/tree/main>

1- follow our algorithm to design test cases and group them in a test case table (chapter 6 of the book). Highlight characteristics, domains, blocks, values. Use the pairwise coverage criteria.

Test Case #	Day	Month	Year	Expected Output	Additional Info
TC1	28	2	1900	1-3-1900	Non-leap year February end
TC2	29	2	2000	1-3-2000	Leap year February end
TC3	30	4	2021	1-5-2021	April end
TC4	31	7	2021	1-8-2021	July end
TC5	31	12	2212	1-1-2213	Year boundary
TC6	15	1	1812	16-1-1812	Normal day
TC7	29	2	1900	Invalid	Non-leap year February
TC8	32	1	2020	Invalid	Invalid day
TC9	31	4	2021	Invalid	Invalid April end
TC10	1	13	2021	Invalid	Invalid month

## **Characteristics and Domains:**

Day= 1-31                      Month= 1-12    Year = 1812-2212

## **Domains and Blocks**

Days valid = 1-28 for all months, 29 (for all months except Feb on non-leap years), 30 (for April, June, September, and November), 31 (for January, March, May, July, August, October, and December)

Days invalid = 0, and all numbers greater than 31

Months valid = 1-12

Months invalid = 0, and all numbers greater than 12

Year valid = 1812-2212

Year invalid = less than 1812, greater than 2212

## **Values for Testing**

Day: 1,15,28,29,30,31

Month: 1,2,4,7,12

Year: 1812, 1900, 2000, 2212

## **Pairwise Coverage**

Pair 1:

Pairs the day and months together to ensure the testing of non-leap years (28 Feb), leap years (29 Feb), as well as the testing of 30 or 31 days in a month (30 April and 31 July)

Pair 2:

Pairs the months and years together to test for Feb. in leap years and non-leap years (Feb 1900 and Feb 2000)

Pair 3:

Pairs the days and years together for boundary testing (31 Dec 2212)

2- write a Junit test script (This is a chance to experiment with different Junit annotations to organize your test cases)

```
1  import static org.junit.jupiter.api.Assertions.assertEquals;
2
3  import org.example.DateCalculator;
4  import org.junit.jupiter.api.BeforeEach;
5  import org.junit.jupiter.api.Test;
6
7  public class NextDateTest {
8
9      // 11 usages
10     private DateCalculator dateCalculator;
11
12     @BeforeEach
13     void setUp() { dateCalculator = new DateCalculator(); }
14
15     @Test
16     void testNextDate_EndOfMonth_NonLeapYear() {
17         assertEquals("1-3-2021", dateCalculator.nextDate(day: 28, month: 2, year: 2021),
18             message: "Testing next day for the end of February in a non-leap year");
19     }
20
21     @Test
22     void testNextDate_EndOfMonth_LeapYear() {
23         assertEquals("29-2-2020", dateCalculator.nextDate(day: 28, month: 2, year: 2020),
24             message: "Testing next day for February 28 in a leap year");
25     }
26
27     @Test
28     void testNextDate_EndOfYear() {
29         assertEquals("1-1-2022", dateCalculator.nextDate(day: 31, month: 12, year: 2021),
30             message: "Testing next day for the end of the year");
31     }
32
33     @Test
34     void testNextDate_MidMonth() {
35         assertEquals("16-7-2021", dateCalculator.nextDate(day: 15, month: 7, year: 2021),
36             message: "Testing a regular day in the middle of a month");
37     }
38
39     @Test
40     void testNextDate_FebruaryNonLeapYear() {
41         assertEquals("Invalid", dateCalculator.nextDate(day: 29, month: 2, year: 1900),
42             message: "Testing invalid date for February 29 in a non-leap year");
43     }
44
45     @Test
46     void testNextDate_InvalidDay() {
47         assertEquals("Invalid", dateCalculator.nextDate(day: 32, month: 1, year: 2020),
48             message: "Testing invalid day");
49     }
50
51     @Test
52     void testNextDate_InvalidAprilEnd() {
53         assertEquals("Invalid", dateCalculator.nextDate(day: 31, month: 4, year: 2021),
54             message: "Testing invalid April end");
55     }
56
57     @Test
58     void testNextDate_InvalidMonth() {
```

```
58     @Test
59     void testNextDate_InvalidMonth() {
60         assertEquals( expected: "Invalid", dateCalculator.nextDate( day: 1, month: 13, year: 2021),
61             message: "Testing invalid month");
62     }
63
64     @Test
65     void testNextDate_NormalDay() {
66         assertEquals( expected: "16-1-1812", dateCalculator.nextDate( day: 15, month: 1, year: 1812),
67             message: "Testing normal day");
68     }
69
70     @Test
71     void testNextDate_YearBoundary() {
72         assertEquals( expected: "1-1-2213", dateCalculator.nextDate( day: 31, month: 12, year: 2212),
73             message: "Testing year boundary");
74     }
75
76     // Add more tests if required for other edge cases or input validations
77 }
78
79
80
```

3- write the corresponding java code

DateCalculator.java

```
1 package org.example;
2
3 3 usages
4 public class DateCalculator {
5     10 usages
6     public String nextDate(int day, int month, int year) {
7         // Check for valid inputs
8         if(day <= 0 || day > 31 || month <= 0 || month > 12 || year < 1812 || year > 2212) {
9             return "Invalid";
10        }
11
12        // Adjust for leap year
13        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
14        int[] daysInMonth = {31, isLeapYear ? 29 : 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
15
16        // Check if the day is valid for the given month
17        if (day > daysInMonth[month - 1]) {
18            return "Invalid";
19        }
20
21        // Increment the day
22        day++;
23
24        if (day > daysInMonth[month - 1]) {
25            day = 1;
26            month++;
27            if (month > 12) {
28                month = 1;
29                year++;
30            }
31        }
32
33        return String.format("%d-%d-%d", day, month, year);
34    }
35}
```

## NextDate.java

```
1 package org.example;
2
3 import java.util.*;
4
5 public class NextDate {
    2 usages
6     public static String calculate(int day, int month, int year) {
7         // Check for valid year
8         if (year < 1812 || year > 2212) {
9             return "Invalid";
10        }
11
12        // Check for valid month
13        if (month < 1 || month > 12) {
14            return "Invalid";
15        }
16
17        // Check for valid day
18        if (day < 1 || day > 31) {
19            return "Invalid";
20        }
21
22        // Handling February and leap years
23        if (month == 2) {
24            if (isLeapYear(year)) {
25                if (day > 29) return "Invalid";
26            } else {
27                if (day > 28) return "Invalid";
28            }
29        }
30
31        // Handling months with 30 days
32        if (Arrays.asList(4, 6, 9, 11).contains(month) && day > 30) {
33            return "Invalid";
34        }
35
36        // Increment the day
37        day++;
38        if (day > 31) {
39            day = 1;
40            month++;
41        }
42
43        if (month == 2 && day > (isLeapYear(year) ? 29 : 28)) {
44            day = 1;
45            month++;
46        }
47
48        if (Arrays.asList(4, 6, 9, 11).contains(month) && day > 30) {
49            day = 1;
50            month++;
51        }
52
53        if (month > 12) {
54            month = 1;
55            year++;
56        }
57    }
```



```

58     return String.format("%d-%d-%d", day, month, year);
59 }
60
61 2 usages
62 private static boolean isLeapYear(int year) {
63     return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
64 }
65
66 public static void main(String[] args) {
67     // The main method can be used to run some basic tests or examples
68     System.out.println(calculate(day: 28, month: 2, year: 2021)); // Outputs "1-3-2021"
69     System.out.println(calculate(day: 29, month: 2, year: 2020)); // Outputs "1-3-2020" (Leap year)
70     // Additional tests can be added here
71 }
72

```

4- run the tests and debug if need be.

5- repeat until the quality of the code is assured

```
C:\Users\Abduh\Desktop\untitled1>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.yourorganization:NextDateProject >-----
[INFO] Building NextDateProject 1.0-SNAPSHOT
[INFO]    from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ NextDateProject ---
[INFO] skip non existing resourceDirectory C:\Users\Abduh\Desktop\untitled1\src\main\resources
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ NextDateProject ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\Abduh\Desktop\untitled1\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ NextDateProject ---
[INFO] skip non existing resourceDirectory C:\Users\Abduh\Desktop\untitled1\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ NextDateProject ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\Abduh\Desktop\untitled1\target\test-classes
[INFO]
[INFO] --- surefire:3.2.2:test (default-test) @ NextDateProject ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running NextDateTest
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.099 s -- in NextDateTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  4.047 s
[INFO] Finished at: 2024-02-28T23:05:49-05:00
[INFO] -----

C:\Users\Abduh\Desktop\untitled1>
```